Introduction :

Le taux de chômage est un indicateur économique essentiel qui reflète la santé du marché du travail au sein d'un pays. Au Maroc, comme dans de nombreux autres pays, le taux de chômage a des implications significatives sur le bien-être socio-économique de la population. Comprendre et anticiper les variations de ce taux est crucial pour élaborer des politiques publiques efficaces, orientées vers la création d'emplois et la stabilisation de l'économie.

Notre projet vise à aborder cette problématique en développant un modèle de prédiction du taux de chômage au Maroc. Pour ce faire, nous adopterons une approche holistique, couvrant plusieurs phases clés du processus de modélisation. Ces étapes comprennent la collecte de données, le prétraitement, la visualisation des données, l'interprétation, le choix des modèles, la construction des modèles, la validation et enfin, l'utilisation des modèles pour générer des prédictions pertinentes.

les bibliographies:

In [168…
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.multioutput import MultiOutputRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.tree import DecisionTreeRegressor
```

Collecte de données:

La première étape de notre projet consiste à rassembler des données pertinentes et fiables liées au marché du travail au Maroc. Cela peut inclure des informations sur l'emploi, le chômage, l'éducation, les secteurs économiques, et d'autres variables pertinentes. La qualité des données est cruciale pour la précision de notre modèle, et nous nous efforcerons d'obtenir des ensembles de données exhaustifs et actualisés.

In [ ]:
```
importation des donnees:
```

In [169…
```
data_age_feminin = pd.read_excel(r"C:\Users\dell\OneDrive\Bureau\age F.xlsx")
data_age_masculin = pd.read_excel(r"C:\Users\dell\OneDrive\Bureau\age mascul.xlsx")
data_dip_feminin = pd.read_excel(r"C:\Users\dell\OneDrive\Bureau\DIPMLOME F.xlsx")
data_dip_masculin = pd.read_excel(r"C:\Users\dell\OneDrive\Bureau\diplome M.xlsx")
```

Prétraitement :

Une fois les données collectées, nous entreprendrons des étapes de prétraitement pour nettoyer et organiser les informations. Cela inclut la gestion des valeurs manquantes, l'élimination des outliers, la normalisation des données, et d'autres techniques visant à rendre les données aptes à être utilisées dans nos modèles de prédiction.

Ajouter la colonne du sex:

In [170…
```
data_age_feminin['sex']='F'
data_age_masculin['sex']='M'
data_dip_feminin['sex']='F'
data_dip_masculin['sex']='M'
```

In [171…
```
data_dip_feminin.tail()
```

Out[171]:

	Années	Sans diplôme	Ayant un diplôme: Niveau moyen	Ayant un diplôme: Niveau supérieur	Ensemble	sex
19	2003	3.971339	30.482607	36.791373	12.393370	F
20	2002	3.836262	29.183204	35.270764	12.122700	F
21	2001	4.063721	30.064752	35.546434	12.200370	F
22	2000	4.339589	33.001484	39.172855	12.775613	F
23	1999	5.459990	34.725906	36.162666	13.159199	F

Regroupement selon l'âge : concaténation des deux sexes dans le même dataframe

In [177…
```
data_age = pd.concat([data_age_feminin, data_age_masculin], axis=0)
data_age.head()
```

Out[177]:

	Années	15 - 24	25 - 34	35 - 44	45 et plus	Ensemble	sex
0	2022	44.4	28.0	9.1	3.2	17.2	F
1	2021	41.9	26.9	9.7	4.0	16.8	F
2	2020	41.2	26.2	8.9	4.0	16.2	F
3	2019	33.4	22.9	6.6	2.3	13.5	F
4	2018	34.3	23.2	7.6	2.3	14.1	F

In [178…
```
data_dip = pd.concat([data_dip_feminin, data_dip_masculin], axis=0)
data_dip.head()
```

Out[178]:

	Années	Sans diplôme	Ayant un diplôme: Niveau moyen	Ayant un diplôme: Niveau supérieur	Ensemble	sex
0	2022	3.8	21.5	34.8	17.2	F
1	2021	4.5	24.5	32.8	16.8	F
2	2020	4.8	24.6	31.8	16.2	F
3	2019	2.9	21.3	29.5	13.5	F
4	2018	3.2	23.2	32.5	14.1	F

Nombre de lignes/colonnes:

In [179…
```
data_age.shape
```

Out[179]:
```
(48, 7)
```

In [180…
```
data_dip.shape
```

Out[180]:
```
(48, 6)
```

In [181…
```
data_age.head()
```

Out[181]:

	Années	15 - 24	25 - 34	35 - 44	45 et plus	Ensemble	sex
0	2022	44.4	28.0	9.1	3.2	17.2	F
1	2021	41.9	26.9	9.7	4.0	16.8	F
2	2020	41.2	26.2	8.9	4.0	16.2	F
3	2019	33.4	22.9	6.6	2.3	13.5	F
4	2018	34.3	23.2	7.6	2.3	14.1	F

In [182…
```
data_dip.head()
```

Out[182]:

	Années	Sans diplôme	Ayant un diplôme: Niveau moyen	Ayant un diplôme: Niveau supérieur	Ensemble	sex
0	2022	3.8	21.5	34.8	17.2	F
1	2021	4.5	24.5	32.8	16.8	F
2	2020	4.8	24.6	31.8	16.2	F
3	2019	2.9	21.3	29.5	13.5	F
4	2018	3.2	23.2	32.5	14.1	F

Renommer les colonnes:

In [183…
```
data_age.columns=['Annes','15-24','25-34','35-44','plus que 45','Ensemble','sex']
```

In [184…
```
data_dip.columns=['Annes','sans d','d moyenne','d superieur','Ensemble','sex']
```

Suppression de colonnes ensembles:

In [185…
```
data_dip = data_dip.drop(columns = ['Ensemble'])
```

In [186…
```
data_age = data_age.drop(columns = ['Ensemble'])
```

Concaténation des deux ensembles de données:

In [222…
```
data_final = pd.concat([data_dip, data_age.drop(['Annes', 'sex'], axis=1)], axis = 1)
data_final.head()
```

Out[222]:

	Annes	sans d	d moyenne	d superieur	sex	15-24	25-34	35-44	plus que 45
0	2022	3.8	21.5	34.8	F	44.4	28.0	9.1	3.2
1	2021	4.5	24.5	32.8	F	41.9	26.9	9.7	4.0
2	2020	4.8	24.6	31.8	F	41.2	26.2	8.9	4.0
3	2019	2.9	21.3	29.5	F	33.4	22.9	6.6	2.3
4	2018	3.2	23.2	32.5	F	34.3	23.2	7.6	2.3

Triage du dataset:

In [188…
```
data_final = data_final.sort_values(by=['Annes', 'sex'], ascending=[False, False], inplace = False)
```

In [189…
```
data_final.shape
```

Out[189]:
```
(48, 9)
```

Les valeurs manquantes:

In [190…
```
data_final.isnull().sum()
```

Out[190]:
```
Annes          0
sans d         0
d moyenne      0
d superieur    0
sex            0
15-24          0
25-34          0
35-44          0
plus que 45    0
dtype: int64
```

Le type de données :

In [191…
```
data_final.dtypes
```

Out[191]:
```
Annes            int64
sans d         float64
d moyenne      float64
d superieur    float64
sex             object
15-24          float64
25-34          float64
35-44          float64
plus que 45    float64
dtype: object
```

Une statistique rapide sur nos données :

In [192…
```
data_final.describe()
```

Out[192]:

	Annes	sans d	d moyenne	d superieur	15-24	25-34	35-44	plus que 45
count	48.000000	48.000000	48.000000	48.000000	48.000000	48.000000	48.000000	48.000000
mean	2010.500000	4.414225	20.729897	24.918585	21.466854	17.083513	6.267704	2.265942
std	6.995439	1.446039	5.944504	7.418594	7.383010	4.272988	1.267009	0.814612
min	1999.000000	2.647612	10.800000	13.964118	14.121230	11.466960	3.600000	0.482458
25%	2004.750000	3.185581	14.750457	17.875000	16.767681	13.142145	5.627507	1.591111
50%	2010.500000	4.219795	21.045649	24.580020	18.800561	16.194735	6.332842	2.183733
75%	2016.250000	5.447532	24.525000	30.300000	22.125000	19.935794	6.794961	2.683076
max	2022.000000	9.131952	34.725906	39.172855	44.400000	28.000000	9.700000	4.000000

Changer les valeurs du sexe par des valeurs numériques:

In [193…
```
data_final['sex'] = data_final['sex'].replace({'M': 1, 'F': -1})
```

In [194…
```
data_final.head()
```

Out[194]:

	Annes	sans d	d moyenne	d superieur	sex	15-24	25-34	35-44	plus que 45
0	2022	4.4	13.0	20.8	1	28.7	16.4	5.6	3.3
0	2022	3.8	21.5	34.8	-1	44.4	28.0	9.1	3.2
1	2021	4.6	14.5	21.9	1	28.4	17.2	6.2	3.7
1	2021	4.5	24.5	32.8	-1	41.9	26.9	9.7	4.0
2	2020	5.8	14.0	19.6	1	28.0	16.2	6.3	3.9

Visualisation des données :

La visualisation des données joue un rôle crucial dans la compréhension des tendances et des relations entre différentes variables. Nous utiliserons des outils graphiques pour représenter les données de manière claire et intuitive, facilitant ainsi l'interprétation des modèles qui seront développés par la suite.

In [195…

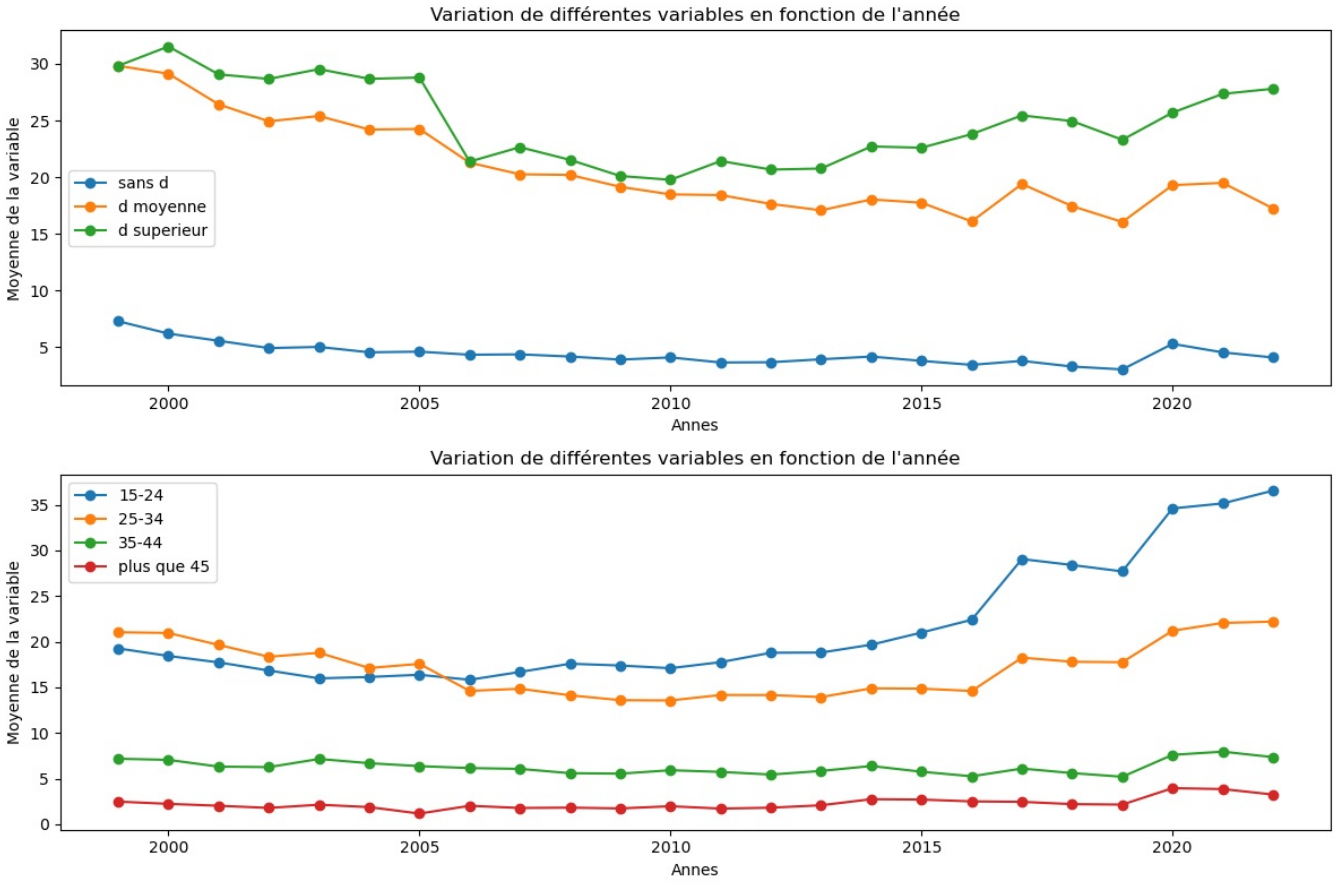
```
target_diplome= ['sans d', 'd moyenne', 'd superieur']
target_age= ['15-24','25-34','35-44','plus que 45']
plt.figure(figsize=(12, 8))
plt.subplot(2, 1, 1)
for i in target_diplome:
        # Group by 'Annes' and calculate mean
        moyenne_par_annees = data_final.groupby('Annes')[i].mean()

        # Plot each line in the same plot
        plt.plot(moyenne_par_annees.index, moyenne_par_annees.values, marker='o', linestyle='-', label=i)

# Add labels and title
plt.xlabel('Annes')
plt.ylabel('Moyenne de la variable')
plt.title("Variation de différentes variables en fonction de l'année")
plt.legend()
plt.subplot(2, 1, 2)
for i in target_age:
        # Group by 'Annes' and calculate mean
        moyenne_par_annees = data_final.groupby('Annes')[i].mean()

        # Plot each line in the same plot
        plt.plot(moyenne_par_annees.index, moyenne_par_annees.values, marker='o', linestyle='-', label=i)
plt.xlabel('Annes')
plt.ylabel('Moyenne de la variable')
plt.title("Variation de différentes variables en fonction de l'année")
# Add legend
plt.legend()

plt.tight_layout()
plt.show()
```



Interprétation des résultats:

Interprétation des résultats:

Taux de chomage selon le diplomes : Le graphe nous montre que le taux de chômage est plus élevé chez les personnes diplômées que chez les personnes non diplômées. En effet, de 1999 à 2005, le taux de chômage chez les personnes diplômées était très élevé, suivi d'une diminution légère . Cependant, à partir de 2020, on observe une reprise de l'augmentation du taux de chômage chez les personnes diplômées. Taux de chomage selon l'age: Le graphique indique que le taux de chômage est plus élevé chez les jeunes que chez les adultes. De 1999 à 2015, le taux de chômage était relativement stable dans toutes les tranches d'âge. Cependant, à partir de 2016, une augmentation progressive s'est manifestée, devenant plus prononcée à partir de 2020. Cette variation du taux de chomage au cours ces années soulève des questions sur les changements socio-économiques ou sur le marché du travail qui pourraient avoir contribué à cette tendance Afin de justifier ces variations, il serait nécessaire d'explorer d'autres facteurs susceptibles d'avoir influencé ces évolutions .Il y a plusieurs facteurs à considérer, mais nous nous avons focalisé sur ceux qui ont une influence majeure et directe sur le taux de chômage, à savoir les épidémies et les investissements de l'État.

In [196…
```
new_features = pd.read_excel(r"C:\Users\dell\OneDrive\Bureau\new_features.xlsx")
```

In [197…
```
new_features.head()
```

Out[197]:

	Annes	effet sanitaire	investissement
0	2022	0.5	1.663588
1	2022	0.5	1.663588
2	2021	3.0	1.596519
3	2021	3.0	1.596519
4	2020	7.0	1.169073

Étant donné que les indices sont différents, l'objectif est de créer les mêmes indices pour les deux dataframes:

In [198…
```
data_final = data_final.reset_index(drop=True)
```

In [199…
```
data_final = pd.concat([data_final, new_features.drop(['Annes'], axis=1)], axis = 1)
```

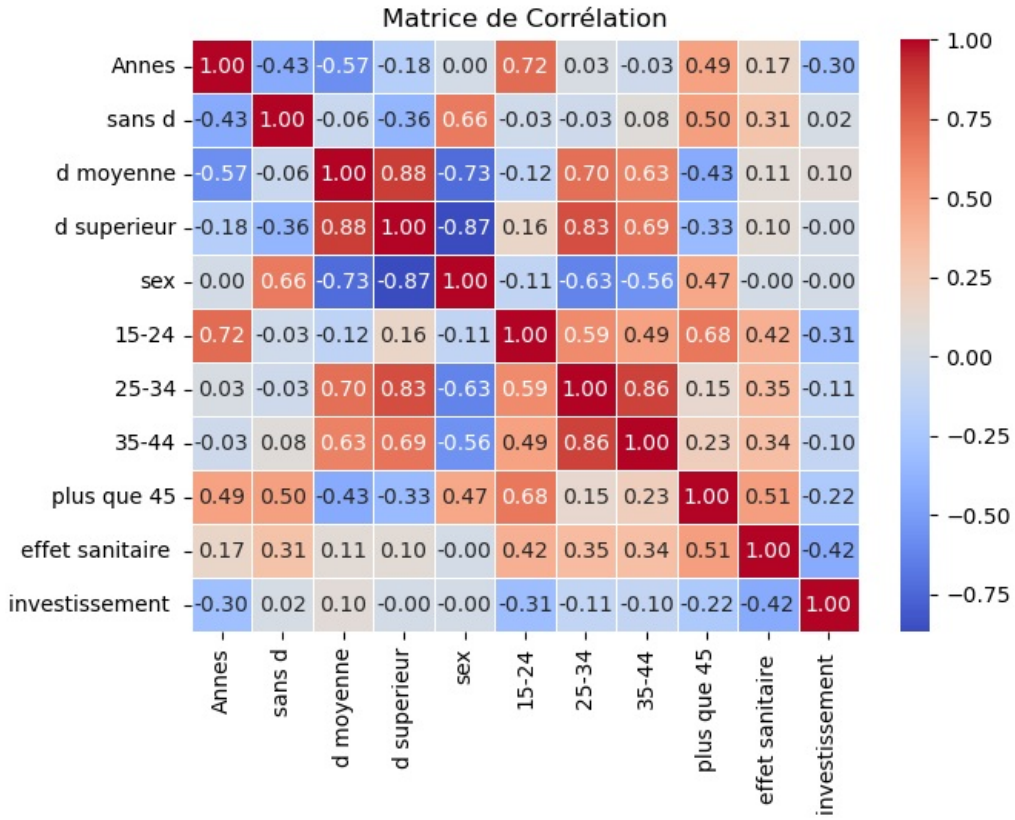
In [200…
```
data_final.head()
```

Out[200]:

	Annes	sans d	d moyenne	d superieur	sex	15-24	25-34	35-44	plus que 45	effet sanitaire	investissement
0	2022	4.4	13.0	20.8	1	28.7	16.4	5.6	3.3	0.5	1.663588
1	2022	3.8	21.5	34.8	-1	44.4	28.0	9.1	3.2	0.5	1.663588
2	2021	4.6	14.5	21.9	1	28.4	17.2	6.2	3.7	3.0	1.596519
3	2021	4.5	24.5	32.8	-1	41.9	26.9	9.7	4.0	3.0	1.596519
4	2020	5.8	14.0	19.6	1	28.0	16.2	6.3	3.9	7.0	1.169073

Corrélation entre les colonnes de notre jeu de données:

In [201…
```
correlation_matrix = data_final.corr()
plt.figure(figsize=(7, 5))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=.5)
plt.title('Matrice de Corrélation')
plt.show()
```



Interprétation des résultats :

En analysant la matrice de corrélation, on peut conclure qu'il existe généralement une faible corrélation linéaire entre la plupart des variables. Cette observation influence significativement la sélection des modèles ultérieurs, car elle suggère que des modèles plus flexibles et capables de capturer des relations non linéaires peuvent être plus appropriés. En d'autres termes, les variables ne semblent pas suivre des tendances simples, et ajuster nos choix de modèles en conséquence nous permettra de mieux représenter la complexité des relations entre elles.

In [202…
```
data_final.head()
```

Out[202]:

	Annes	sans d	d moyenne	d superieur	sex	15-24	25-34	35-44	plus que 45	effet sanitaire	investissement
0	2022	4.4	13.0	20.8	1	28.7	16.4	5.6	3.3	0.5	1.663588
1	2022	3.8	21.5	34.8	-1	44.4	28.0	9.1	3.2	0.5	1.663588
2	2021	4.6	14.5	21.9	1	28.4	17.2	6.2	3.7	3.0	1.596519
3	2021	4.5	24.5	32.8	-1	41.9	26.9	9.7	4.0	3.0	1.596519
4	2020	5.8	14.0	19.6	1	28.0	16.2	6.3	3.9	7.0	1.169073

Variation de la colonne cible en fonction des années.

In [203…
```
data_final.columns=['Annes','sans d','d moyenne','d superieur', 'sex', '15-24','25-34','35-44','plus que 45','e
```

In [204…
```
data_final.head()
```

Out[204]:

	Annes	sans d	d moyenne	d superieur	sex	15-24	25-34	35-44	plus que 45	effet sanitaire	investissement
0	2022	4.4	13.0	20.8	1	28.7	16.4	5.6	3.3	0.5	1.663588
1	2022	3.8	21.5	34.8	-1	44.4	28.0	9.1	3.2	0.5	1.663588
2	2021	4.6	14.5	21.9	1	28.4	17.2	6.2	3.7	3.0	1.596519
3	2021	4.5	24.5	32.8	-1	41.9	26.9	9.7	4.0	3.0	1.596519
4	2020	5.8	14.0	19.6	1	28.0	16.2	6.3	3.9	7.0	1.169073

Pour appréhender l'essor marqué du chômage en 2020 et 1999, une approche pertinente serait d'élaborer des représentations graphiques illustrant les tendances des facteurs sanitaires et des investissements au fil du temps. En esquissant ces courbes, notre objectif est d'analyser de manière approfondie l'impact des variables liées à la santé et aux investissements, fournissant ainsi un éclairage précis sur les moteurs sous-jacents de la montée du chômage pendant ces deux années cruciales. À travers cette visualisation, nous cherchons à décoder les dynamiques complexes qui ont exercé une influence significative sur le paysage de l'emploi au cours de ces périodes spécifiques.

In [205…
```
features = ['investissement', 'effet sanitaire']
```

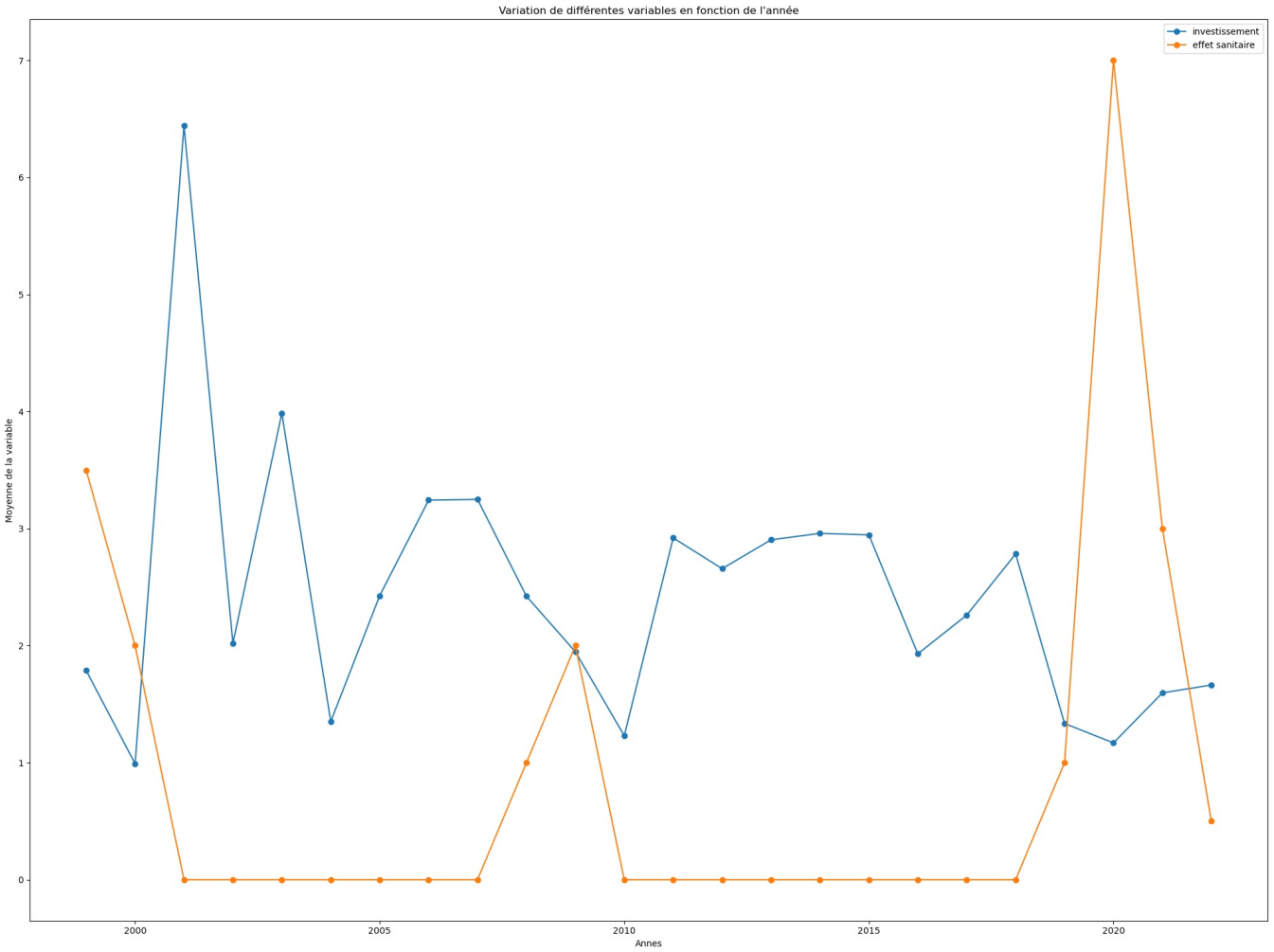
```
plt.figure(figsize=(20, 15))
for i in features:
        # Group by 'Annes' and calculate mean
        moyenne_par_annees = data_final.groupby('Annes')[i].mean()

        # Plot each line in the same plot
        plt.plot(moyenne_par_annees.index, moyenne_par_annees.values, marker='o', linestyle='-', label=i)

# Add labels and title
plt.xlabel('Annes')
plt.ylabel('Moyenne de la variable')
plt.title("Variation de différentes variables en fonction de l'année")

# Add legend
plt.legend()

plt.tight_layout()
plt.show()
```



Interprétation des résultats:

D'après ce graphe, on peut conclure que les investissements de l'État pour réduire le taux de chômage ont un impact remarquable. Le taux de chômage a diminué au cours des années où l'État a créé un nombre significatif d'offres d'emplois. En revanche, les épidémies ont une influence majeure sur le taux de chômage, comme en témoigne la crise du coronavirus en 2020, où le taux de chômage a atteint son sommet.

In [206…
```
data_final.head()
```

Out[206]:

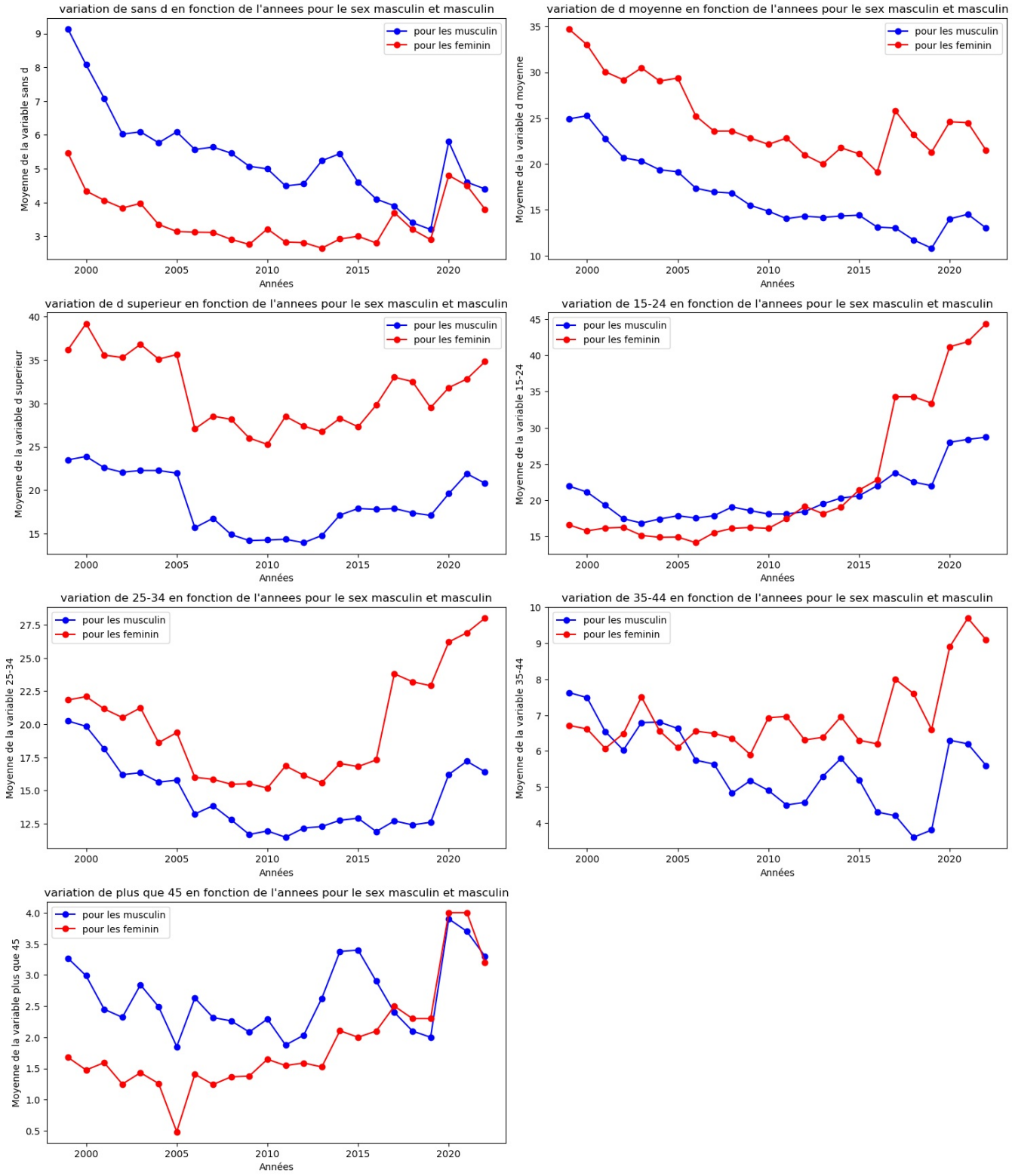
	Annes	sans d	d moyenne	d superieur	sex	15-24	25-34	35-44	plus que 45	effet sanitaire	investissement
0	2022	4.4	13.0	20.8	1	28.7	16.4	5.6	3.3	0.5	1.663588
1	2022	3.8	21.5	34.8	-1	44.4	28.0	9.1	3.2	0.5	1.663588
2	2021	4.6	14.5	21.9	1	28.4	17.2	6.2	3.7	3.0	1.596519
3	2021	4.5	24.5	32.8	-1	41.9	26.9	9.7	4.0	3.0	1.596519
4	2020	5.8	14.0	19.6	1	28.0	16.2	6.3	3.9	7.0	1.169073

maintenet on trace la courbe de variation de taux de chomage pour les deux sexe:

In [207…
```
#variation de taux de chomage par sex feminin su fil des annees
```

```
plt.figure(figsize = (15, 30))
y=1
for i in target:
        moyeen_par_annees_sexe = data_final.groupby(['sex', 'Annes'])[i].mean()
        par_sexe_M = moyeen_par_annees_sexe[1]
        par_sexe_F = moyeen_par_annees_sexe[-1]

        plt.subplot(7,2,y)
        plt.plot(par_sexe_M.index, par_sexe_M.values, marker='o', linestyle='-', color='b', label='pour les mus
        plt.plot(par_sexe_F.index, par_sexe_F.values, marker='o', linestyle='-', color='r', label='pour les fem
        plt.legend()
        plt.xlabel('Années')
        plt.ylabel('Moyenne de la variable {}'.format(i))
        plt.title("variation de {} en fonction de l'annees pour le sex masculin et masculin".format(i))
        y+=1
plt.tight_layout()
```



Interprétation des résultats:

L'analyse des sous-graphiques suggère que, globalement, la variation du taux de chômage semble suivre une tendance similaire pour les sexes masculin et féminin au fil des années. Cela suggère que, dans l'ensemble, les facteurs affectant le chômage semblent

influencer de manière cohérente les deux sexes, indépendamment du niveau d'éducation ou de l'âge.

Cependant, une observation importante ressort de manière constante : la moyenne du taux de chômage pour la population féminine est systématiquement supérieure à celle de la population masculine. Cette disparité persistante indique qu'il existe probablement des facteurs spécifiques au genre qui contribuent à des taux de chômage plus élevés chez les femmes, malgré des tendances similaires de variation au fil des années.

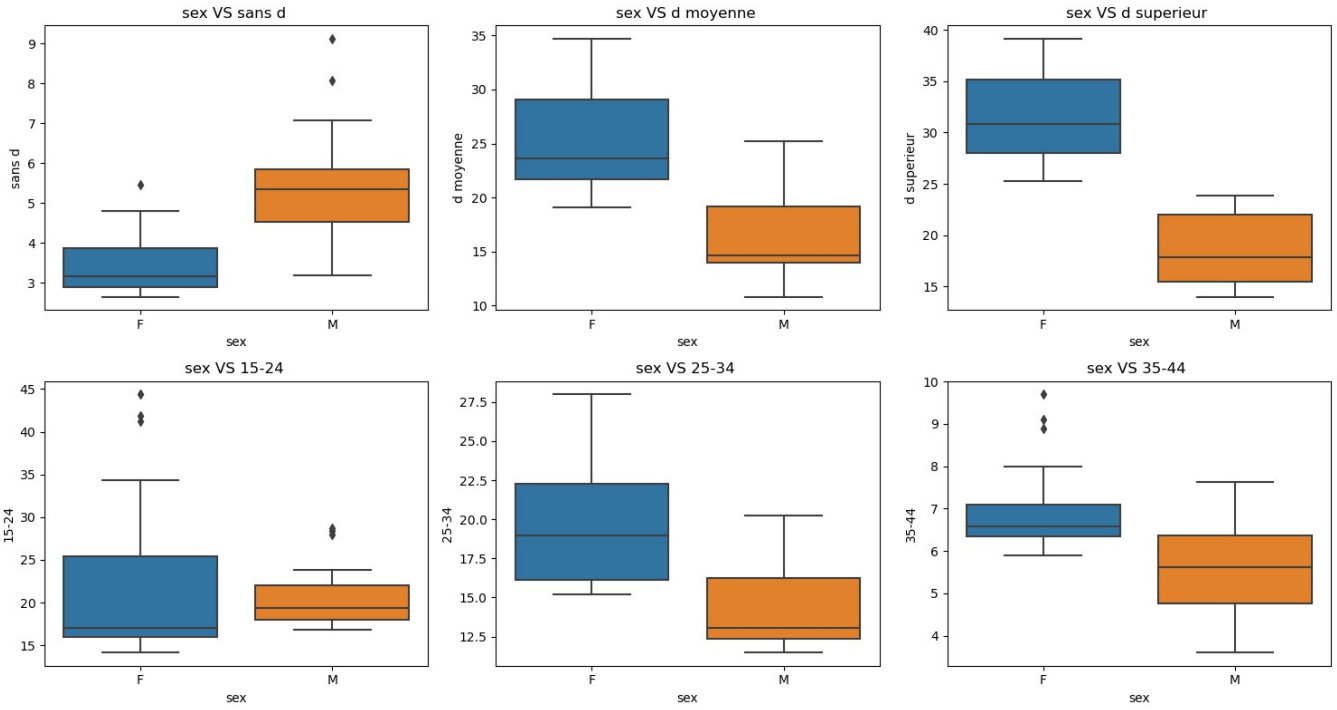
Il est crucial de creuser davantage pour comprendre les raisons derrière cette disparité de genre dans le taux de chômage, car cela pourrait être lié à des dynamiques spécifiques sur le marché du travail ou à des défis auxquels les femmes sont confrontées dans le contexte économique étudié. Cette observation souligne l'importance de politiques visant à réduire les disparités de genre sur le marché du travail et à promouvoir l'égalité des opportunités professionnelles pour les hommes et les femmes.donne moi une resme de cette paragraphe en six lignes pa plus pas moins.

In [225…
```
target = ['sans d', 'd moyenne', 'd superieur', '15-24', '25-34', '35-44', 'plus que 45']

fig, ax = plt.subplots(2, 3, figsize=(15, 8))

y = 0
for i in range(2):
    for j in range(3):
        if y < len(target):
            sns.boxplot(x='sex', y=target[y], data=data_final, ax=ax[i, j]).set_title('sex VS {} '.format(targe
            y += 1

plt.tight_layout()
plt.show()
```



Une grille de boxplots est une représentation graphique qui permet de visualiser la distribution des données pour plusieurs variables ou catégories u ainsi les écartilles, le médian et le min et le max valeur . en remarque que le diplôme supérieur est le plus élevé, suivi par ceux ayant une diplôme moyenne.

Standardisation des données:

In [209…
```
features = ['Annes', 'sex', 'effet sanitaire', 'investissement']

scaler = StandardScaler()

data_final[features] = scaler.fit_transform(data_final[features])
```

In [210…
```
data_final.head()
```

Out[210]:

	Annes	sans d	d moyenne	d superieur	sex	15-24	25-34	35-44	plus que 45	effet sanitaire	investissement
0	1.661325	4.4	13.0	20.8	1.0	28.7	16.4	5.6	3.3	-0.204390	-0.672202
1	1.661325	3.8	21.5	34.8	-1.0	44.4	28.0	9.1	3.2	-0.204390	-0.672202
2	1.516862	4.6	14.5	21.9	1.0	28.4	17.2	6.2	3.7	1.328538	-0.731341
3	1.516862	4.5	24.5	32.8	-1.0	41.9	26.9	9.7	4.0	1.328538	-0.731341
4	1.372399	5.8	14.0	19.6	1.0	28.0	16.2	6.3	3.9	3.781223	-1.108249

Division de la base de données en données d'entraînement (train) et de test (test):

In [211…
```
features = ['Annes', 'sex', 'effet sanitaire', 'investissement']
target = ['sans d','d moyenne','d superieur', '15-24','25-34','35-44','plus que 45']

X_train, X_test, y_train, y_test = train_test_split(data_final[features], data_final[target], test_size=0.2, ra
```

Prediction du taux de chomage au Maroc

choix de modele :

on vas utiliser Decision Tree Regressor, Random Forest Regressor et Gradient Boosting Regressor pour predire le taux de chomage au maroc a couse des raison suivantes :

Problème de régression : Les données que vous traitez sont continues ou proches les unes des autres, ce qui indique un problème de régression plutôt qu'un problème de classification. Les modèles de régression sont appropriés pour prédire des valeurs continues.

Non-linéarité des relations : Il est souligné qu'il n'y a pas de relation linéaire entre la majorité des colonnes de votre ensemble de données. Cela justifie l'utilisation de modèles de régression non linéaires, tels que les arbres de décision, les forêts aléatoires et les boosters.

Nature complexe et non linéaire des données : La complexité et la non-linéarité des données sont des facteurs clés qui motivent le choix de modèles robustes et performants. Les modèles tels que les forêts aléatoires et les boosters sont capables de capturer des relations complexes et de fournir des prédictions précises.

MultiOutputRegressor : La cible comprenant plusieurs colonnes nécessite l'utilisation de la fonction MultiOutputRegressor. Cela permet d'adapter le modèle à plusieurs sorties simultanément, ce qui est approprié dans le cas où vous essayez de prédire plusieurs variables cibles.

Construction et validaion des modeles:

Randomforest

In [212…
```
param = {
    'estimator__n_estimators': [1000, 3000],
    'estimator__max_depth': [None, 20],
    'estimator__random_state': [42, 567]
}


grid = GridSearchCV(MultiOutputRegressor(estimator=RandomForestRegressor()), param_grid=param, scoring='r2', cv

grid.fit(X_train, y_train)

best_params = grid.best_params_
print("Meilleurs hyperparamètres :", best_params)
best_model = grid.best_estimator_

# Faire des prédictions avec le meilleur modèle
y_train_pred = best_model.predict(X_train)
y_test_pred = best_model.predict(X_test)

# Évaluer la performance
train_r2 = r2_score(y_train, y_train_pred)
test_r2 = r2_score(y_test, y_test_pred)
print('Train R² score:', train_r2)
print('Test R² score:', test_r2)
```

```
Meilleurs hyperparamètres : {'estimator__max_depth': None, 'estimator__n_estimators': 3000, 'estimator__random_
state': 42}
Train R² score: 0.9729358598351627
Test R² score: 0.8648293675052423
```

Les résultats fournis pour RandomForestRegressor comprennent les meilleurs hyperparamètres sélectionnés par la recherche sur grille (GridSearchCV) : * estimator__max_depth : None --> La profondeur maximale des arbres est réglée sur "Aucune limite", ce qui signifie que les arbres peuvent se développer jusqu'à ce que chaque feuille contienne un seul point ou qu'un autre critère d'arrêt soit atteint. * estimator__n_estimators : 3000 --> Le nombre d'arbres dans l'ensemble est réglé sur 3000, indiquant la taille de l'ensemble d'arbres. * estimator__random_state : 42 --> Le générateur de nombres aléatoires est fixé à 42 pour assurer la reproductibilité des résultats. R squared score sur le data du training 0.9729 et R squared score sur le data du test est 0.8648 ce qui montrant une excellente performance sur l'ensemble d'entraînement et une bonne capacité de généralisation sur l'ensemble de test

Decision Trees:

In [166…
```
decision_tree =DecisionTreeRegressor(random_state=5)
param_grid = {
```

```
    'max_depth': [k for k in range(1,100,10)],
    'min_samples_split': [k for k in range(2,6)],
    'min_samples_leaf': [k for k in range(1,5)]
}


grid_search = GridSearchCV(decision_tree, param_grid, cv=5, scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)

print('Les meilleurs paramètres sont:', grid_search.best_params_)

best_model = grid_search.best_estimator_
best_model.fit(X_train,y_train)
x=best_model.score(X_test,y_test)
y=best_model.score(X_train,y_train)
print('test score : ', x, 'train score', y)
```

```
Les meilleurs paramètres sont: {'max_depth': 11, 'min_samples_leaf': 1, 'min_samples_split': 4}
test score :  0.8747152427732499 train score 0.9642488228751727
```

Les résultats fournis pour DecisionTreeRegressor comprennent les meilleurs hyperparamètres sélectionnés par la recherche sur grille (GridSearchCV) : * max_depth : 11 --> La profondeur maximale de l'arbre de décision est fixée à 11, ce qui limite la croissance de l'arbre. * min_samples_leaf : 1 --> Le nombre minimum d'échantillons requis pour être dans une feuille est fixé à 1, indiquant que chaque feuille peut contenir un seul échantillon. * min_samples_split : 4 --> Le nombre minimum d'échantillons requis pour diviser un nœud est fixé à 4. mean square error score sur le data du training 0.96424 et sur le data du test est 0.8747 ce qui montrant une excellente performance sur l'ensemble d'entraînement et une bonne capacité de généralisation sur l'ensemble de test

GradientBoostingRegressor

In [213…

```
param = {
    'estimator__n_estimators': [50, 100, 200],
    'estimator__learning_rate': [0.01, 0.1, 0.2],
    'estimator__max_depth': [3, 4, 5],
}

grid = GridSearchCV(MultiOutputRegressor(estimator=GradientBoostingRegressor()), param_grid=param, scoring='r2'

grid.fit(X_train, y_train)

best_params = grid.best_params_
print("Meilleurs hyperparamètres :", best_params)
best_model = grid.best_estimator_

# Faire des prédictions avec le meilleur modèle
y_train_pred = best_model.predict(X_train)
y_test_pred = best_model.predict(X_test)

# Évaluer la performance
train_r2 = r2_score(y_train, y_train_pred)
test_r2 = r2_score(y_test, y_test_pred)
print('Train R² score:', train_r2)
print('Test R² score:', test_r2)
```

```
Meilleurs hyperparamètres : {'estimator__learning_rate': 0.01, 'estimator__max_depth': 3, 'estimator__n_estimat
ors': 200}
Train R² score: 0.9479313453909388
Test R² score: 0.8452260052968742
```

voici une cas d'utilisations on va predire le taux de chomage pour 2030 dans les conditions suivant: evenement mondial coup du monde alors les investissment augmenteront jusqu'à 8.66 milliard et le maroc a sortie d'une crise sanitaire depuis 2 ans avant 2030

In [217…

```
new_data={'Annes':2030,'sex':[1,-1],'effet sanitaire':2, 'investissement':8.66}
new_data=pd.DataFrame(new_data,columns=['Annes', 'sex', 'effet sanitaire', 'investissement'])
new_data
```

Out[217]:

	Annes	sex	effet sanitaire	investissement
0	2030	1	2	8.66
1	2030	-1	2	8.66

In [218…

```
taux_chomage = best_model.predict(new_data)
taux_chomage=pd.DataFrame(taux_chomage,columns=['sans d','d moyenne','d superieur', '15-24','25-34','35-44','pl
taux_chomage
```

Out[218]:

	sans d	d moyenne	d superieur	15-24	25-34	35-44	plus que 45
0	5.677926	14.216952	20.372110	25.621112	16.345246	6.027122	3.412469
1	5.071961	23.086519	32.316922	40.754667	26.282883	8.455695	3.278156

In [223…

```
print("2022\n",data_final.loc[data_final['Annes']==2022,['sans d','d moyenne','d superieur', '15-24','25-34','3
```

```
2022
    sans d  d moyenne  d superieur  15-24  25-34  35-44  plus que 45
0      3.8       21.5         34.8   44.4   28.0    9.1          3.2
0      4.4       13.0         20.8   28.7   16.4    5.6          3.3
```

principe de GridSearchCv

GridSearchCV est une technique d'optimisation d'hyperparamètres qui permet de rechercher la meilleure combinaison d'hyperparamètres pour un modèle donné. Le principe de GridSearchCV est le suivant :

1. Définition des hyperparamètres à tester : Vous spécifions une grille (ou un ensemble) d'hyperparamètres que vous souhaitez optimiser. Cela peut inclure des paramètres tels que la profondeur maximale d'un arbre de décision, le nombre d'estimateurs dans un modèle d'ensemble, le taux d'apprentissage dans un algorithme de gradient boosting, etc.

2.Choix du modèle : Nous sélectionnons le modèle que Nous souhaitons entraîner et optimiser. Cela peut être n'importe quel modèle de scikit-learn qui expose des paramètres à ajuster.

1. Validation croisée : Nous spécifiez le nombre de plis pour la validation croisée (cv), qui divise votre ensemble de données en plusieurs parties. Le modèle sera entraîné sur certaines parties et évalué sur d'autres. Cela aide à obtenir une évaluation plus robuste des performances du modèle.
2. Création de la grille de recherche : GridSearchCV crée toutes les combinaisons possibles des hyperparamètres que vous avez spécifiés dans la grille. Chaque combinaison est utilisée pour entraîner et évaluer le modèle.
3. Évaluation de la performance : Le modèle est entraîné et évalué pour chaque combinaison d'hyperparamètres à l'aide de la validation croisée. La métrique spécifiée (par exemple, précision, F1-score, erreur quadratique moyenne, etc.) est utilisée pour évaluer la performance.
4. Meilleurs hyperparamètres : Une fois que toutes les combinaisons ont été évaluées, GridSearchCV identifie la combinaison d'hyperparamètres qui a produit les meilleures performances selon la métrique spécifiée.
5. Entraînement du modèle final :
    - Nous pouvons utiliser les meilleurs hyperparamètres pour entraîner le modèle sur l'ensemble complet de données (ou sur un ensemble d'entraînement particulier) afin d'obtenir le modèle final.

    GridSearchCV simplifie le processus d'exploration des hyperparamètres en automatisant la recherche exhaustive, permettant ainsi de trouver la meilleure configuration d'hyperparamètres pour maximiser les performances du modèle. Cependant, cela peut être coûteux en termes de temps de calcul, surtout si la grille de recherche est grande.

Remarque nous pouvons utiliser aussi la courbe de selectionne ou bien des boucle repititive pour obtenire des matrices carres contiens des scors pour chaque model lié par un certains hyperparamétre et on prendre la moyenne plus elevé

Gradient boosting regressor

Cet exemple de code utilise la recherche par grille (GridSearchCV) avec la classe MultiOutputRegressor pour optimiser les hyperparamètres d'un modèle de régression par Gradient Boosting (GradientBoostingRegressor) , alors on va expliquer chaque partie de ce code :

1.Définition des hyperparamètres à optimiser :

- n_estimators: Le nombre d'arbres dans le modèle. Plus le nombre est élevé, plus le modèle est complexe.
- learning_rate: Le taux d'apprentissage contrôle la contribution de chaque arbre au modèle. Une valeur plus basse nécessite un nombre plus élevé d'arbres pour atteindre la même complexité.
- max_depth : La profondeur maximale de chaque arbre. Contrôle la complexité de chaque arbre.

1. Création de la grille de recherche (param_grid) :
    - La grille de recherche spécifie toutes les combinaisons possibles des valeurs d'hyperparamètres à tester.
2. Création de l'estimateur (GradientBoostingRegressor) :
    - La classe MultiOutputRegressor est utilisée pour traiter plusieurs sorties simultanément. Dans cet exemple, elle est appliquée à un GradientBoostingRegressor.
3. Création de l'objet GridSearchCV :
    - L'objet GridSearchCV prend l'estimateur, la grille d'hyperparamètres, la métrique de performance (scoring), et le nombre de plis pour la validation croisée (cv).

5.Exécution de la recherche par grille :

- GridSearchCV ajuste le modèle pour chaque combinaison d'hyperparamètres et évalue la performance à l'aide de la validation

croisée.

1. Identification des meilleurs hyperparamètres :
    - Une fois la recherche par grille terminée, best*params* contient les valeurs d'hyperparamètres qui ont donné les meilleurs résultats.
2. Entraînement du meilleur modèle :
    - Le meilleur modèle est ensuite ajusté sur l'ensemble d'entraînement complet avec les hyperparamètres optimaux.
3. Prédictions et évaluation de la performance :
    - Le modèle est utilisé pour faire des prédictions sur les ensembles d'entraînement (y_train_pred) et de test (y_test_pred).
    - La performance du modèle est évaluée à l'aide du coefficient de détermination (R^2) sur les ensembles d'entraînement et de test.
4. Affichage des résultats :
    - Les résultats finaux, y compris les meilleurs hyperparamètres et les scores R^2, sont affichés.

Cet exemple illustre comment utiliser la recherche par grille pour ajuster les hyperparamètres d'un modèle de Gradient Boosting avec prise en charge de multiples sorties grâce à MultiOutputRegressor.

In [ ]: